In the Claims:

1-11. (Cancelled)

12. (Original) A method comprising:

raising a current priority of a task to a current priority of a higher priority task when the higher priority task blocks on a resource held by the task;

incrementing a priority inheritance variable when the higher priority task blocks on the resource held by the task, the priority inheritance variable associated with the task and configured to be indicative of the number of resources held by the task that higher priority tasks are waiting to receive;

decrementing the priority inheritance variable when the task releases the resource that the higher priority task has blocked on;

testing the priority inheritance variable; and

lowering the current priority of the task when testing the priority inheritance variable indicates that the task holds no resources that are involved in a priority inheritance.

13. (Original) A method comprising:

testing a priority inheritance variable associated with a task, the priority inheritance variable configured to have a value indicative of the number of inversion safe mutual exclusion semaphores held by the task that higher priority tasks are waiting to receive; and

setting a current priority of the task to a base priority value when testing the priority inheritance variable indicates that no higher priority tasks are waiting to receive inversion safe mutual exclusion semaphores held by the task.

14. (Original) The method of claim 13, further comprising:

raising the current priority of the task when a higher priority task blocks on an inversion safe mutual exclusion semaphore held by the task.

15. (Original) The method of claim 14, wherein, when raising the current priority of the task when the higher priority task blocks on the inversion safe mutual exclusion semaphore held by

the task, the current priority of the task is raised to a current priority of the higher priority task that blocked on the inversion safe mutual exclusion semaphore held by the task.

16. (Original) The method of claim 13, further comprising:

adjusting the priority inheritance variable when a higher priority task blocks on an inversion safe mutual exclusion semaphore held by the task.

17. (Original) The method of claim 16, wherein, when adjusting the priority inheritance variable when the higher priority blocks on the inversion safe mutual exclusion semaphore held by the task, the priority inheritance variable is incremented.

18. (Original) The method of claim 13, further comprising:

adjusting the priority inheritance variable when the task releases an inversion safe mutual exclusion semaphore.

19. (Original) The method of claim 18, wherein,

when adjusting the priority inheritance variable when the task releases the inversion safe mutual exclusion semaphore, the priority inheritance variable is decremented.

20. (Original) The method of claim 13, wherein the priority inheritance variable is included in a task control block for the task.

21. (Previously presented) A method comprising:

raising a current priority of a task when a higher priority task blocks on a first inversion safe mutual exclusion semaphore, the first inversion safe mutual exclusion semaphore being held by the task;

incrementing a counter when the higher priority task blocks on the first inversion safe mutual exclusion, the counter associated with the task and configured to have a value indicative of the number of inversion safe mutual exclusion semaphores held by the task that higher priority

tasks are waiting to receive;

decrementing the counter when the task releases the first inversion safe mutual exclusion semaphore; and

while the task still holds a second inversion safe mutual exclusion semaphore, setting the current priority of the task to a base priority value when the counter is decremented to a value that indicates that the task holds no inversion safe mutual exclusion semaphores involved in priority inheritance.

22 - 28. (Cancelled)

29. (Currently amended) ~~The system according to claim 26, further~~ A system comprising:

a semaphore;

a variable, the variable associated with the semaphore and configured to indicate whether a pending request for the semaphore has resulted in a priority inheritance; and

a task holding the semaphore,

wherein the variable is configured to indicate a count of the number of pending requests for the semaphore made by tasks with higher priority than the task holding the semaphore.

30. (Cancelled)

31. (Original) A method comprising:

tracking a number of inversion safe mutual exclusion semaphores held by a task that higher priority tasks are presently blocked on, the tracking using only a predetermined amount of memory;

raising a current priority of the task when a higher priority task blocks on an inversion safe mutual exclusion semaphore held by the task; and

setting the current priority of the task to a base priority value whenever no higher priority tasks are waiting to receive any of the inversion safe mutual exclusion semaphores held by the task and the task still holds at least one inversion safe mutual exclusion semaphore.

32. (Cancelled)

33. (Original) An article of manufacture comprising a computer-readable medium having stored thereon instructions adapted to be executed by a processor, the instructions which, when executed, define a series of steps to be used to control priority inheritance, said steps comprising:

raising a current priority of a task when a higher priority task blocks on an inversion safe mutual exclusion semaphore, the inversion safe mutual exclusion semaphore being held by the task;

incrementing a counter when the higher priority task blocks on the inversion safe mutual exclusion, the counter associated with the task and configured to have a value indicative of the number of inversion safe mutual exclusion semaphores held by the task that higher priority tasks are waiting to receive;

decrementing the counter when the task releases the inversion safe mutual exclusion semaphore; and

setting the current priority of the task to a base priority value when the counter is decremented to a value that indicates that the task holds no inversion safe mutual exclusion semaphores involved in priority inheritance.

34. (Original) An article of manufacture comprising a computer-readable medium having stored thereon instructions adapted to be executed by a processor, the instructions which, when executed, define a series of steps to be used to control priority inheritance, said steps comprising:

tracking a number of resources held by a task that higher priority tasks are presently blocked on, the tracking using only a predetermined amount of memory;

raising a current priority of the task when a higher priority task blocks on a resource held by the task; and

setting the current priority of the task to a base priority value whenever no higher priority tasks are waiting to receive any of the resources held by the task and the task still holds at least one resource.